import java.util.Scanner;

import java.io.File;

/\*\*

\*

\* @author Samuel Swedberg

\* @version 9/30/22

\*

\* The Recursion class holds algorithms to be called by the Client.

\*

\*/

public class Recursion {

Scanner sc = new Scanner(System.in);

int count = 0;

/\*\*

\* Algorithm 1

\* @param n

\* @return

\*/

public double algorithm1(int n)

{

if(n == 1.0)

return 1.0;

else

return (1.0 / n) + algorithm1(n - 1); // Harmonic recursion

}

/\*\*

\* Algorithm 2

\* @param n

\* @return

\*/

public double algorithm2(int n)

{

if(n == 1.0)

return 1.0;

else

return n + algorithm1(n-1); // Sum of positive integers recursion

}

/\*\*

\* Algorithm 3

\* @param targetFileName

\* @param startPath

\*/

public void findFile( String targetFileName, String startPath)

{

File file = new File(startPath);

File[] files = file.listFiles();

// Runs only if files exist

if(files != null)

{

// For loop to run through all files

for(int i=0; i<files.length; i++)

{

File f = files[i];

// Checks if a directory or file

if(f.isDirectory())

findFile(targetFileName, f.getAbsolutePath()); // findFile recursion

// Checks if targetFileName is the same as f.getName()

if(f.getName().equals(targetFileName))

{

count++; // Increments entries found

System.out.printf("Algorithm3( %s, %s ) = targetFileName: " + f.getName() + " in " + f.getAbsolutePath() + "\n", targetFileName, startPath);

}

}

}

return;

}

}

import java.util.Scanner;

import java.io.File;

import java.io.FileNotFoundException;

/\*\*

\*

\* @author Samuel Swedberg

\* @version 9/30/22

\*

\* A client that demonstrates recursion algorithms.

\*

\*/

public class Client {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

boolean quit = false;

// While loop for the Menu

while (!quit) {

System.out.println("========== Menu ==========");

System.out.println(" A) Algorithm 1 ");

System.out.println(" B) Algorithm 2 ");

System.out.println(" C) Algorithm 3 ");

System.out.println(" Q) Quit ");

System.out.print("Enter choice: ");

String choice = sc.nextLine();

switch (choice) {

case "A":

case "a":

callAlgorithm1();

break;

case "B":

case "b":

callAlgorithm2();

break;

case "C":

case "c":

callAlgorithm3();

break;

case "Q":

case "q":

quit = true;

break;

default:

System.out.println("Invalid Choice, Try Again\n");

}

}

}

/\*\*

\* Calls Algorithm1 from Recursion class

\*/

public static void callAlgorithm1()

{

Recursion rc = new Recursion();

Scanner sc = new Scanner(System.in);

boolean tryAgain = true;

int n = 0;

String answer = "";

// Loops until user wants to quit

while(tryAgain)

{

boolean inValidInput = true;

// Loops until valid input is found

while(inValidInput)

{

System.out.println("Algorithm1> Type in an integer greater than 0: ");

answer = sc.nextLine();

try

{

n = Integer.parseInt(answer);

if(n < 0)

throw new IllegalArgumentException("Algorithm1> Number must be greater than 0.");

inValidInput = false;

// NumberFormatException

} catch ( NumberFormatException nfe ) {

System.out.println("Algorithm1> You must enter an integer value. Press a key to continue or N to leave: ");

answer = sc.nextLine();

// Asks user if they want to continue

if(answer.equalsIgnoreCase("N"))

tryAgain = false;

break;

// IllegalArgumentException

} catch (IllegalArgumentException iae) {

System.out.println(iae.getMessage() + " Press a key to continue or N to leave: ");

answer = sc.nextLine();

// Asks user if they want to continue

if(answer.equalsIgnoreCase("N"))

tryAgain = false;

break;

}

}

// Runs only if valid input is found

if(!inValidInput)

{

System.out.printf("Algorithm1( %d ) = %,f \n ", n, rc.algorithm1(n));

System.out.println("Press a key to continue or N to leave: ");

answer = sc.nextLine();

// Asks user if they want to continue

if(answer.equalsIgnoreCase("N"))

tryAgain = false;

}

}

}

/\*\*

\* Calls Algorithm2 from Recursion class

\*/

public static void callAlgorithm2()

{

Recursion rc = new Recursion();

Scanner sc = new Scanner(System.in);

boolean tryAgain = true;

int n = 0;

String answer = "";

// Loops until user wants to quit

while(tryAgain)

{

boolean inValidInput = true;

// Loops until valid input is found

while(inValidInput)

{

System.out.println("Algorithm2> Type in an integer greater than 0: ");

answer = sc.nextLine();

try

{

n = Integer.parseInt(answer);

if(n < 0)

throw new IllegalArgumentException("Algorithm2> Number must be greater than 0.");

inValidInput = false;

// NumberFormatException

} catch ( NumberFormatException nfe ) {

System.out.println("Algorithm2> You must enter an integer value. Press a key to continue or N to leave: ");

answer = sc.nextLine();

// Asks user if they want to continue

if(answer.equalsIgnoreCase("N"))

tryAgain = false;

break;

// IllegalArgumentException

} catch (IllegalArgumentException iae) {

System.out.println(iae.getMessage() + " Press a key to continue or N to leave: ");

answer = sc.nextLine();

// Asks user if they want to continue

if(answer.equalsIgnoreCase("N"))

tryAgain = false;

break;

}

}

// Runs only if valid input is found

if(!inValidInput)

{

System.out.printf("Algorithm2( %d ) = %,f \n ", n, rc.algorithm2(n));

System.out.println("Press a key to continue or N to leave: ");

answer = sc.nextLine();

// Asks user if they want to continue

if(answer.equalsIgnoreCase("N"))

tryAgain = false;

}

}

}

/\*\*

\* Calls Algorithm3 from Recursion class

\*/

public static void callAlgorithm3()

{

Recursion rc = new Recursion();

Scanner sc = new Scanner(System.in);

boolean tryAgain = true;

int n = 0;

String directory = "", filename = "", answer = "";

// Loops until user wants to quit

while(tryAgain)

{

rc.count = 0;

boolean inValidInput = true;

//Loops until valid input is found

while(inValidInput)

{

System.out.println("Algorithm3> Enter starting directory: ");

directory = sc.nextLine();

try {

if(!new File(directory).exists())

throw new FileNotFoundException("Algorithm3> You must only enter a directory.");

inValidInput = false;

// FileNotFoundException

} catch (FileNotFoundException fnfe) {

System.out.println(fnfe.getMessage() + " Press a key to continue or N to leave: ");

answer = sc.nextLine();

// Asks user if they want to continue

if(answer.equalsIgnoreCase("N"))

tryAgain = false;

break;

}

}

// Runs only if valid input is found

if(!inValidInput)

{

System.out.println("Algorithm3> Enter filename: ");

filename = sc.nextLine();

rc.findFile(filename, directory);

System.out.printf("Algorithm3> Found %d entries\n", rc.count);

System.out.println("Press a key to continue or N to leave: ");

answer = sc.nextLine();

// Asks user if they want to continue

if(answer.equalsIgnoreCase("N"))

tryAgain = false;

}

}

}

}

run:

========== Menu ==========

A) Algorithm 1

B) Algorithm 2

C) Algorithm 3

Q) Quit

Enter choice: a

Algorithm1> Type in an integer greater than 0:

2

Algorithm1( 2 ) = 1.500000

Press a key to continue or N to leave:

Algorithm1> Type in an integer greater than 0:

4

Algorithm1( 4 ) = 2.083333

Press a key to continue or N to leave:

n

========== Menu ==========

A) Algorithm 1

B) Algorithm 2

C) Algorithm 3

Q) Quit

Enter choice: b

Algorithm2> Type in an integer greater than 0:

3

Algorithm2( 3 ) = 4.500000

Press a key to continue or N to leave:

Algorithm2> Type in an integer greater than 0:

4

Algorithm2( 4 ) = 5.833333

Press a key to continue or N to leave:

n

========== Menu ==========

A) Algorithm 1

B) Algorithm 2

C) Algorithm 3

Q) Quit

Enter choice: 3

Invalid Choice, Try Again

========== Menu ==========

A) Algorithm 1

B) Algorithm 2

C) Algorithm 3

Q) Quit

Enter choice: c

Algorithm3> Enter starting directory:

A:\

Algorithm3> Enter filename:

Recovery.txt

Algorithm3( Recovery.txt, A:\ ) = targetFileName: Recovery.txt in A:\Recovery.txt

Algorithm3> Found 1 entries

Press a key to continue or N to leave:

Algorithm3> Enter starting directory:

d

Algorithm3> You must only enter a directory. Press a key to continue or N to leave:

n

========== Menu ==========

A) Algorithm 1

B) Algorithm 2

C) Algorithm 3

Q) Quit

Enter choice: q

BUILD SUCCESSFUL (total time: 1 minute 21 seconds)